Analysis of Algorithms
CS 372

Dynamic programming

(Based on slides by M.Nicolescu)

Dynamic Programming

- An algorithm design technique (like divide and conquer)
- Divide and conquer
  - Partition the problem into independent subproblems
  - Solve the subproblems recursively
  - Combine the solutions to solve the original problem

Dynamic Programming

- Applicable when subproblems are not independent
  - Subproblems share subsubproblems
  - A divide and conquer approach would repeatedly solve the common subproblems
  - Dynamic programming solves every subproblem just once and stores the answer in a table

3

Dynamic Programming

- Used for **optimization problems**
  - A set of choices must be made to get an optimal solution
  - Find a solution with the optimal value (minimum or maximum)
  - There may be many solutions that return the optimal value: **an optimal solution**

4

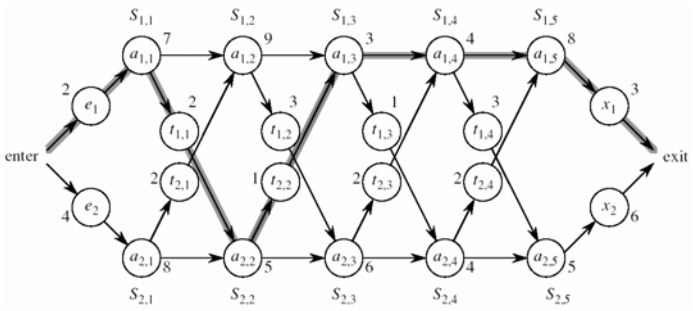
Dynamic Programming Algorithm

1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution in a bottom-up fashion
4. Construct an optimal solution from computed information

5

Assembly Line Scheduling

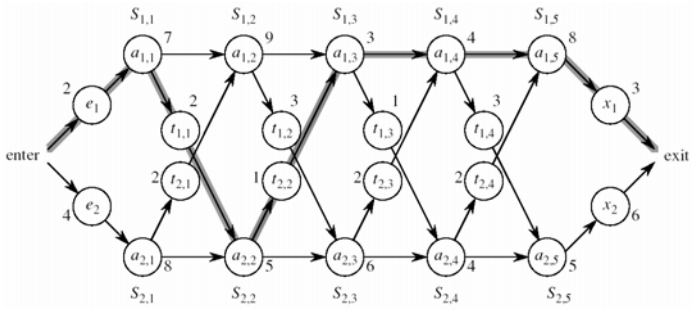
- Automobile factory with two assembly lines
  - Each line has n stations: $S_{1,1}, \ldots, S_{1,n}$ and $S_{2,1}, \ldots, S_{2,n}$
  - Corresponding stations $S_{1,j}$ and $S_{2,j}$ perform the same function but can take different amounts of time $a_{1,j}$ and $a_{2,j}$
  - Entry times e_1 and e_2 and exit times x_1 and x_2



6

Assembly Line

- After going through a station, can either:
  - stay on same line at no cost, or
  - transfer to other line: cost after $S_{i,j}$ is $t_{i,j}$, $j = 1, \ldots, n - 1$

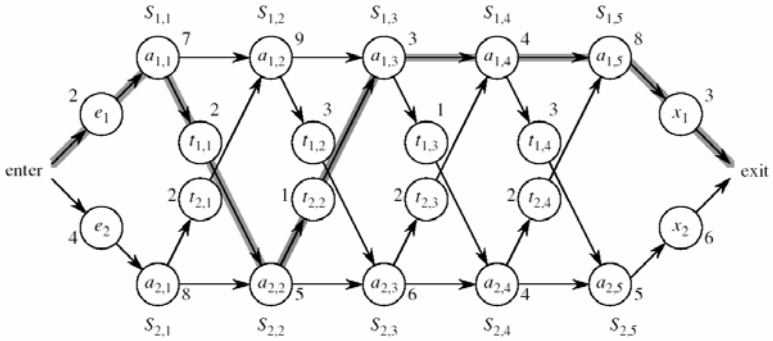


7

Assembly Line Scheduling

- Problem:

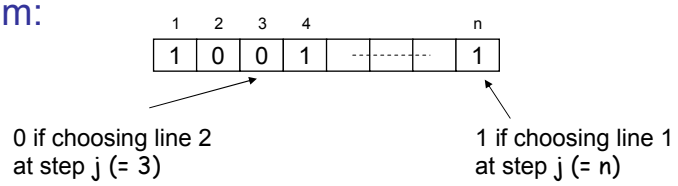
  what stations should be chosen from line 1 and which from line 2 in order to minimize the total time through the factory for one car?



8

One Solution

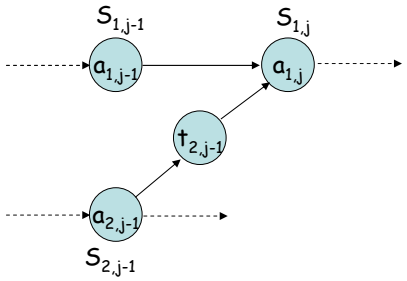
- Brute force
  - Enumerate all possibilities of selecting stations
  - Compute how long it takes in each case and choose the best one
- Problem:



  - There are 2^n possible ways to choose stations
  - Infeasible when n is large

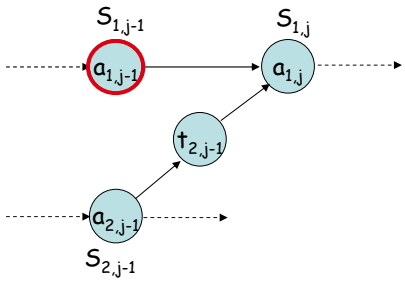
1. Structure of the Optimal Solution

- Let's consider all possible ways to get from the starting point through station $S_{1,j}$
  - We have two choices of how to get to $S_{1,j}$:
    - Through $S_{1,j-1}$, then directly to $S_{1,j}$
    - Through $S_{2,j-1}$, then transfer over to $S_{1,j}$



1. Structure of the Optimal Solution

- Suppose that the fastest way through $S_{1,j}$ is through $S_{1,j-1}$
  - We must have taken a fastest way from entry through $S_{1,j-1}$
  - If there were a faster way through $S_{1,j-1}$, we would use it instead
- Similarly for $S_{2,j-1}$



11

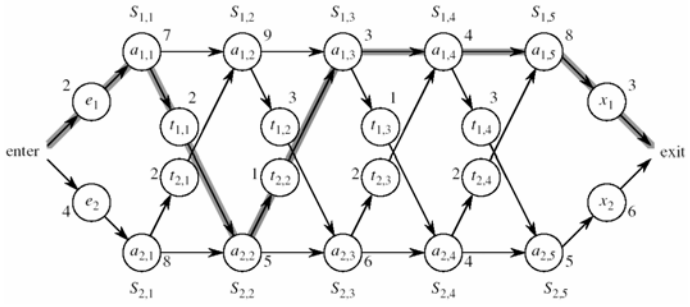
Optimal Substructure

- **Generalization**: an optimal solution to the problem *find the fastest way through* $S_{1,j}$ contains within it an optimal solution to subproblems: *find the fastest way through* $S_{1,j-1}$ *or* $S_{2,j-1}$.
- This is referred to as the **optimal substructure** property
- We use this property to construct an optimal solution to a problem from optimal solutions to subproblems

12

2. A Recursive Solution

- Define the value of an optimal solution in terms of the optimal solution to subproblems
- Assembly line subproblems
  - Finding the fastest way through station j on both lines, j = 1, 2, …, n

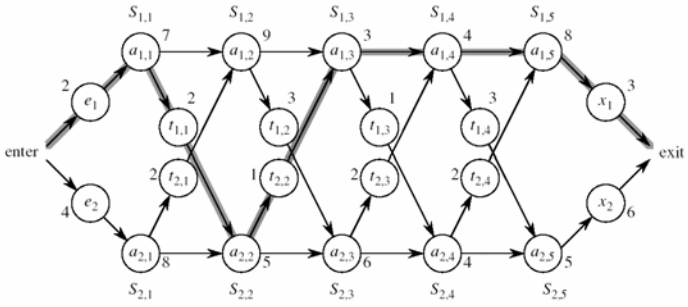


13

2. A Recursive Solution

- f^* = the fastest time to get through the entire factory
- $f_i[j]$ = the fastest time to get from the starting point through station $S_{i,j}$

$f^* = \min(f_1[n] + x_1, f_2[n] + x_2)$



14

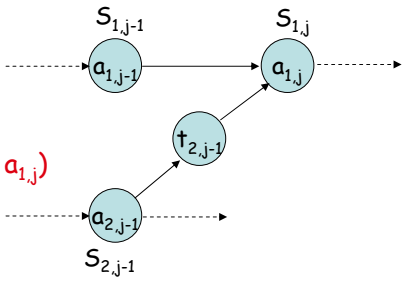
2. A Recursive Solution (cont.)

- Compute $f_i[j]$ for j = 2, 3, …,n, and i = 1, 2
- Fastest way through $S_{1,j}$ is either:
  - the way through $S_{1,j-1}$ then directly through $S_{1,j}$, or

    $f_1[j - 1] + a_{1,j}$
  - the way through $S_{2,j-1}$, transfer from line 2 to line 1, then through $S_{1,j}$

    $f_2[j -1] + t_{2,j-1} + a_{1,j}$

$f_1[j] = \min(f_1[j - 1] + a_{1,j}, f_2[j -1] + t_{2,j-1} + a_{1,j})$



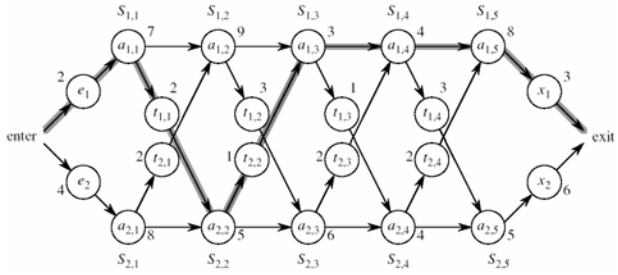
15

2. A Recursive Solution (cont.)

- $f_i[j]$ = the fastest time to get from the starting point through station $S_{i,j}$
- j = 1 (getting through station 1)

  $f_1[1] = e_1 + a_{1,1}$

  $f_2[1] = e_2 + a_{2,1}$



16

2. A Recursive Solution (cont.)

$$f_1[j] = \begin{cases} e_1 + a_{1,1} & \text{if } j = 1 \\ \min(f_1[j-1] + a_{1,j},\ f_2[j-1] + t_{2,j-1} + a_{1,j}) & \text{if } j \geq 2 \end{cases}$$

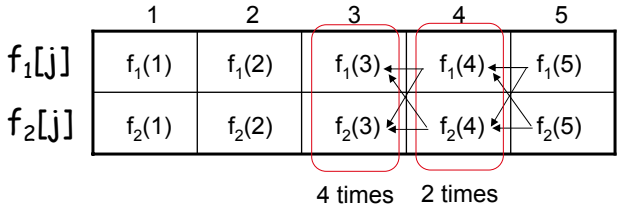
$$f_2[j] = \begin{cases} e_2 + a_{2,1} & \text{if } j = 1 \\ \min(f_2[j-1] + a_{2,j},\ f_1[j-1] + t_{1,j-1} + a_{2,j}) & \text{if } j \geq 2 \end{cases}$$

17

3. Computing the Optimal Solution

$f^* = \min(f_1[n] + x_1,\ f_2[n] + x_2)$

$f_1[j] = \min(f_2[j-1] + a_{2,j},\ f_1[j-1] + t_{1,j-1} + a_{2,j})$

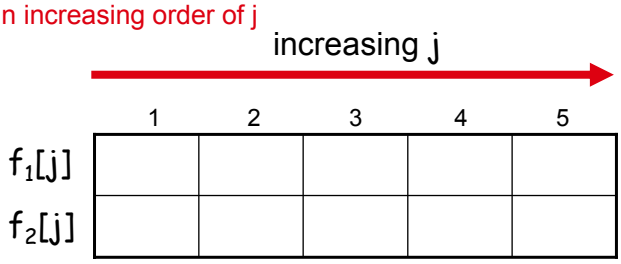


- Solving top-down would result in exponential running time

18

3. Computing the Optimal Solution

- For $j \geq 2$, each value $f_i[j]$ depends only on the values of $f_1[j-1]$ and $f_2[j-1]$
- Compute the values of $f_i[j]$
  - in increasing order of j



- Bottom-up approach
  - First find optimal solutions to subproblems
  - Find an optimal solution to the problem from the subproblems

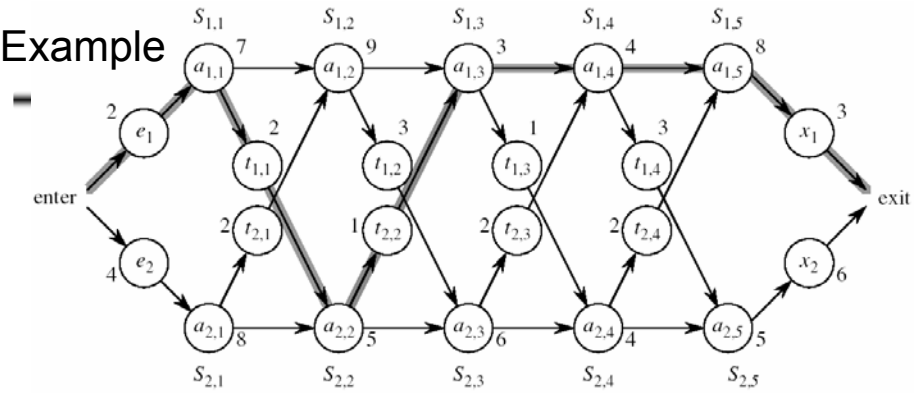
19

Construct the Optimal Solution

- We need the sequence of what line has been used at each station:
  - $l_i[j]$ – the line number (1, 2) whose station $(j-1)$ has been used to get in fastest time through $S_{i,j}$, $j = 2, 3, \ldots, n$
  - l^* - the line whose station n is used to get in the fastest way through the entire factory

20

Example



$$f_1[j] = \begin{cases} e_1 + a_{1,1}, & \text{if } j = 1 \\ \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j}) & \text{if } j \geq 2 \end{cases}$$

	1	2	3	4	5
$f_1[j]$	9	$18^{[1]}$	$20^{[2]}$	$24^{[1]}$	$32^{[1]}$
$f_2[j]$	12	$16^{[1]}$	$22^{[2]}$	$25^{[1]}$	$30^{[2]}$

$f^* = 35^{[1]}$

FASTEST-WAY(a, t, e, x, n)

1. $f_1[1] \leftarrow e_1 + a_{1,1}$
2. $f_2[1] \leftarrow e_2 + a_{2,1}$

} Compute initial values of f_1 and f_2

3. **for** $j \leftarrow 2$ **to** n
4. **do if** $f_1[j-1] + a_{1,j} \leq f_2[j-1] + t_{2,j-1} + a_{1,j}$
5. **then** $f_1[j] \leftarrow f_1[j-1] + a_{1,j}$
6. $l_1[j] \leftarrow 1$
7. **else** $f_1[j] \leftarrow f_2[j-1] + t_{2,j-1} + a_{1,j}$
8. $l_1[j] \leftarrow 2$

} Compute the values of $f_1[j]$ and $l_1[j]$

9. **if** $f_2[j-1] + a_{2,j} \leq f_1[j-1] + t_{1,j-1} + a_{2,j}$
10. **then** $f_2[j] \leftarrow f_2[j-1] + a_{2,j}$
11. $l_2[j] \leftarrow 2$
12. **else** $f_2[j] \leftarrow f_1[j-1] + t_{1,j-1} + a_{2,j}$
13. $l_2[j] \leftarrow 1$

} Compute the values of $f_2[j]$ and $l_2[j]$

FASTEST-WAY(a, t, e, x, n) (cont.)

```
14.  if f1[n] + x1 ≤ f2[n] + x2
15.     then f* = f1[n] + x1
16.          l* = 1
17.     else f* = f2[n] + x2
18.          l* = 2
```

Compute the values of the fastest time through the entire factory

4. Construct an Optimal Solution

Alg.: PRINT-STATIONS(l, n)

```
i ← l*
print "line " i ", station " n
for j ← n downto 2
    do i ← li[j]
    print "line " i ", station " j - 1
```

line 1, station 5

line 1, station 4

line 1, station 3

line 2, station 2

line 1, station 1

	1	2	3	4	5
$f_1[j]/l_1[j]$	9	18[1]	20[2]	24[1]	32[1]
$f_2[j]/l_2[j]$	12	16[1]	22[2]	25[1]	30[2]

l* = 1

Dynamic Programming Algorithm

1. Characterize the structure of an optimal solution
   - Fastest time through a station depends on the fastest time on previous stations
2. Recursively define the value of an optimal solution
   - $f_1[j] = \min(f_1[j - 1] + a_{1,j}, f_2[j - 1] + t_{2,j-1} + a_{1,j})$
3. Compute the value of an optimal solution in a bottom-up fashion
   - Fill in the fastest time table in increasing order of j (station #)
4. Construct an optimal solution from computed information
   - Use an additional table to help reconstruct the optimal solution

25